Summary of INF5620

Xing Cai

Nov 18, 2016

Disclaimer

The following slides are only meant as a very superficial overview of the content of the INF5620 course. **Students are strongly recommended to read the lecture notes and study guide slides for details.**

Scalar ODEs

A simple exponential decay model

$$u'(t) = -au(t), \quad u(0) = I,\ t \in (0, T],$$

where $a > 0$ is a constant.

A simple vibration model

$$u''(t) + \omega^2 u(t) = 0, \quad u(0) = I,\ u'(0) = 0,\ t \in (0, T]$$

Finite difference discretization

1. discretizing the domain (time for ODEs, time/space for PDEs),
2. fulfilling the equation at discrete mesh points,
3. replacing derivatives by finite differences,
4. formulating a difference equation.

FDM for solving a scalar ODE

The time domain $[0, T]$ is represented by a *mesh*: a finite number of $N_t + 1$ points

$$0 = t_0 < t_1 < t_2 < \cdots < t_{N_t - 1} < t_{N_t} = T$$

- We seek the solution u at the mesh points: $u(t_n)$, $n = 1, 2, \ldots, N_t$.
- Note: u^0 is known as I.
- Notational short-form for the numerical approximation to $u(t_n)$: u^n

Approximating u' by finite differences

Forward Euler method

$$u'(t_n) \approx \frac{u^{n+1} - u^n}{t_{n+1} - t_n}$$

Backward Euler method

$$u'(t_n) \approx \frac{u^n - u^{n-1}}{t_n - t_{n-1}}$$

Crank-Nicolson method

$$u'(t_{n-1} + \frac{t_n - t_{n-1}}{2}) \approx \frac{u^n - u^{n-1}}{t_n - t_{n-1}}$$

Approximating u'' by centered finite difference

Assuming a uniform time step size Δt:

$$u''(t_n) \approx \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2}$$

Examples of the resulting numerical schemes

Forward Euler scheme for the exponential decay model

$$u^{n+1} = (1 - a(t_{n+1} - t_n))\, u^n$$

Backward Euler scheme for the exponential decay model

$$u^{n+1} = \frac{1}{1 + a(t_{n+1} - t_n)} u^n$$

Crank-Nicolson scheme for the exponential decay model

$$u^{n+1} = \frac{1 - \frac{1}{2}a(t_{n+1} - t_n)}{1 + \frac{1}{2}a(t_{n+1} - t_n)} u^n$$

Finite difference scheme for the vibration model

First step:

$$u^1 = u^0 - \frac{1}{2}\Delta t^2\omega^2 u^0$$

For later steps ($n \geq 1$):

$$u^{n+1} = 2u^n - u^{n-1} - \Delta t^2\omega^2 u^n$$

Questions

- How accurate is a numerical scheme (with respect to Δt)?
- Is there any limit on the size of Δt?

Example: θ-rule for the decay model

$$u^{n+1} = \frac{1-(1-\theta)a\Delta t}{1+\theta a\Delta t}u^n$$

$\theta = 0$: Forward Euler, $\theta = 1$: Backward Euler, $\theta = \frac{1}{2}$: Crank-Nicolson

- The exact solution is known to be monotonly decaying
- How will the numerical solutions depend on Δt and θ?

Simple stability analysis

$$u^n = IA^n, \quad A = \frac{1-(1-\theta)a\Delta t}{1+\theta a\Delta t}$$

- When $\theta = 1$, we always have $0 < A < 1$ independent of Δt
- When $\theta = 0$, we need $\Delta < 1/a$ to ensure $0 < A < 1$
- When $\theta = \frac{1}{2}$, we need $\Delta < 2/a$ to ensure $0 < A < 1$

Simple accuracy analysis

The exact solution of the exponential decay model is

$$u_e(t) = Ie^{-at}$$

Therefore, the true error of the numerical solution at $t = t_n$ is

$$Ie^{-at_n} - IA^n$$

Taylor series expansion reveals that the true error is $\mathcal{O}(\Delta t)$ for Backward and Forward Euler schemes, $\mathcal{O}(\Delta t^2)$ for Crank-Nicolson.

Analysis for the vibration model

The difference equation

$$u^{n+1} = 2u^n - u^{n-1} - \Delta t^2 \omega^2 u^n$$

is linear and homogeneous, thus $u^n \sim IA^n$ where A is to be determined.

Exact solution of the vibration model $\sim I\exp(i\omega t) = I\left(e^{i\omega\Delta t}\right)^n$, we expect $A = \exp(i\tilde{\omega}\Delta t)$. Inserting $u^n = IA^n$ into the difference equation, we can arrive at

$$\frac{4}{\Delta t^2}\sin^2(\frac{\tilde{\omega}\Delta t}{2}) = \omega^2$$

Analysis for the vibration model (2)

$$\tilde{\omega} = \pm\frac{2}{\Delta t}\sin^{-1}\left(\frac{\omega\Delta t}{2}\right)$$

- $\tilde{\omega} \neq \omega$
- $\tilde{\omega} - \omega$ is the frequency error
- To ensure the numerical solution having a constant amplitude (as the exact solution) it requires $\sin^{-1}(\omega\Delta t/2)$ to be real-valued $\Rightarrow |\omega\Delta t/2| \leq 1 \Rightarrow \Delta t \leq \frac{2}{\omega}$

Convergence

A numerical scheme is convergent if the error goes to zero when the discretization parameter (such as Δt) goes to zero.

Truncation error

We can insert the exact solution u_e into the discrete equation of a numerical scheme, and see how well u_e fits the discrete equation. The "residual" is the truncation error.

Please read the module on this particular topic for more info.

Simple PDEs

A simple 1D diffusion model

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad x \in (0, L),\ t \in (0, T] \tag{1}$$

$$u(x, 0) = I(x), \qquad x \in [0, L] \tag{2}$$

$$u(0, t) = 0, \quad u(L, t) = 0, \qquad t > 0, \tag{3}$$

A simple 1D wave model

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \qquad x \in (0, L),\ t \in (0, T] \tag{4}$$

$$u(x, 0) = I(x), \qquad x \in [0, L] \tag{5}$$

$$\frac{\partial}{\partial t} u(x, 0) = 0, \qquad x \in [0, L] \tag{6}$$

$$u(0, t) = 0, \quad u(L, t) = 0, \qquad t \in (0, T] \tag{7}$$

FDM: Forward Euler scheme for 1D diffusion

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{8}$$

FDM: Backward Euler scheme for 1D diffusion

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{9}$$

FDM: Crank-Nicolson scheme for 1D diffusion

$$u_i^{n+1} - \frac{1}{2}F(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}) = u_i^n + \frac{1}{2}F(u_{i-1}^n - 2u_i^n + u_{i+1}^n) \quad (10)$$

$F = \alpha \frac{\Delta t}{\Delta x^2}$ is known as the *mesh Fourier number*.

FDM: Centered differences for 1D wave

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + C^2\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \tag{11}$$

$C = c\frac{\Delta t}{\Delta x}$ is known as the (dimensionless) *Courant number*

Explicit vs. implicit discretization methods

- Explicit discretization methods:
  - Forward-Euler scheme for the diffusion equation
  - Centered finite difference scheme for the wave equation
- Implicit discretization methods:
  - Backward-Euler scheme for the diffusion equation
  - Crank-Nicolson scheme for the diffusion equation

Implicit discretization methods for PDEs have to solve systems of linear algebraic equations!

Example: backward-Euler method for 1D diffusion

$$-Fu^n_{i-1} + (1+2F)\,u^n_i - Fu^n_{i+1} = u^{n-1}_{i-1} \tag{12}$$

for $i = 1, \ldots, Nx - 1$.

What are the unknowns in the linear system?

1. either u^n_i for $i = 1, \ldots, N_x - 1$ (all *internal* spatial mesh points)
2. or u^n_i, $i = 0, \ldots, N_x$ (all spatial points)

The linear system in matrix notation:

$$AU = b, \quad U = (u^n_0, \ldots, u^n_{N_x})$$

A is very sparse

$$A = \begin{pmatrix} A_{0,0} & A_{0,1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ A_{1,0} & A_{1,1} & A_{1,2} & \ddots & & & & & \vdots \\ 0 & A_{2,1} & A_{2,2} & A_{2,3} & \ddots & & & & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 0 & A_{i,i-1} & A_{i,i} & A_{i,i+1} & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & \ddots & \ddots & \ddots & A_{N_x-1,N} \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & A_{N_x,N_x-1} & A_{N_x,N_x} \end{pmatrix} \tag{13}$$

Detailed expressions for the matrix entries

The nonzero entries are given by

$$A_{i,i-1} = -F \tag{14}$$
$$A_{i,i} = 1 + 2F \tag{15}$$
$$A_{i,i+1} = -F \tag{16}$$

for $i = 1, \ldots, N_x - 1$.

The equations for the boundary points correspond to

$$A_{0,0} = 1, \quad A_{0,1} = 0, \quad A_{N_x,N_x-1} = 0, \quad A_{N_x,N_x} = 1$$

The right-hand side

$$b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_i \\ \vdots \\ b_{N_x} \end{pmatrix} \tag{17}$$

with

$$b_0 = 0 \tag{18}$$

$$b_i = u_i^{n-1}, \quad i = 1, \ldots, N_x - 1 \tag{19}$$

$$b_{N_x} = 0 \tag{20}$$

Accuracy and stability analysis

Fourier analysis is an important tool

Please read the corresponding parts in the lecture notes for details.

Another approach to approximating functions/equations

- FDM: finding u_i as approximate of $u_e(x_i)$, for $i = 0, 1, \ldots, N$
- Another approach:

$$u(x) = \sum_{j=0}^{N} c_j \psi_j(x)$$

such that $u \in V$, which is a given function space spanned by basis functions $\psi_0, \psi_1, \ldots, \psi_N$. (The degrees of freedom are the scalar coefficients $c_0, c_1, \ldots, c_N$.)

Approximating a function $f(x)$

Idea: find $c_0, c_1, \ldots, c_N$ such that $u(x) = \sum_{j=0}^{N} c_j \psi_j(x)$ is a "best" approximation to $f(x)$. (Approximation error: $f(x) - u(x)$)

- Least squares method: minimization of the square norm of the error, i.e., $(e, e) = \int_\Omega e(x)e(x)\,dx$.

$$\sum_{j\in\mathcal{I}_s} A_{i,j}c_j = b_i,\ i \in \mathcal{I}_s, \quad A_{i,j} = (\psi_i, \psi_j),\ b_i = (f, \psi_i)$$

- Projection/Galerkin method: make the error $f - u$ orthogonal to V. Same result as the least squares method
- Collocation/interpolation method: force $u(x_i) = f(x_i)$ at some selected *collocation* points $\{x_i\}_{i\in\mathcal{I}_s}$.

$$\sum_{j\in\mathcal{I}_s} A_{i,j}c_j = b_i,\ i \in \mathcal{I}_s, \quad A_{i,j} = \psi_j(x_i),\ b_i = f(x_i)$$

"Quality" of the basis functions $\psi_0, \psi_1, \ldots, \psi_N$

- For the least squares or projection/Galerkin methods, the ideal scenario is to have a set of orthogonal basis functions, $(\psi_i, \psi_j) = 0$ when $i \neq j$.
- However, it is not easy to design othorgonal basis functions for any (arbitrary) domain in higher space dimensions
- Idea: *Local support*: $\psi_i(x) \neq 0$ for x in a small subdomain of Ω

Finite element basis functions

Split Ω into N_e non-overlapping subdomains called *elements*:

$$\Omega = \Omega^{(0)} \cup \cdots \cup \Omega^{(N_e)}$$

On each element, introduce N_n points called *nodes*: $x_0, \ldots, x_{N_n-1}$

- The finite element basis functions are named $\varphi_i(x)$
- $\varphi_i = 1$ at node i and 0 at all other nodes
- φ_i is a Lagrange polynomial or 0 on each element
- For nodes at the boundary between two elements, φ_i is made up of two Lagrange polynomials, one over each element

Recall: Lagrange polynomials

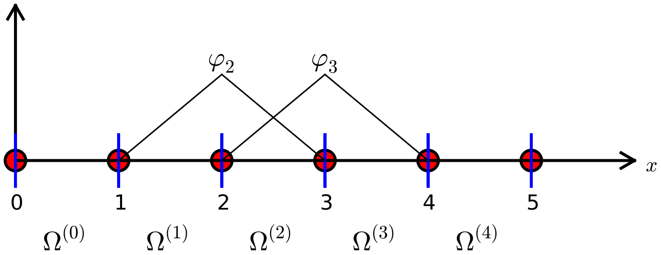
Given $x_0, x_1, \ldots, x_N$

$$\psi_i(x) = \prod_{j=0, j\neq i}^{N} \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_N}{x_i - x_N}$$

Nice property:

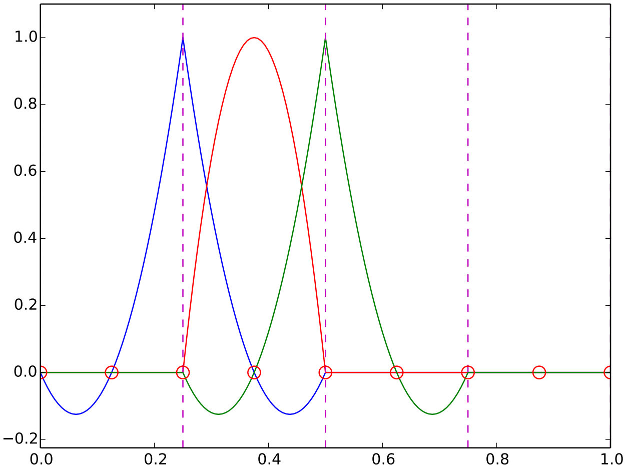
$$\psi_i(x_j) = \delta_{ij}, \quad \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

P1 elements



φ_2
φ_3
x
0
1
2
3
4
5
$\Omega^{(0)}$
$\Omega^{(1)}$
$\Omega^{(2)}$
$\Omega^{(3)}$
$\Omega^{(4)}$

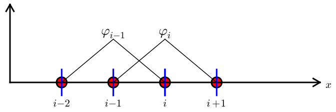
P2 elements



Split the integrals into elementwise integrals

$$A_{i,j} = \int_\Omega \varphi_i\varphi_j dx = \sum_e \int_{\Omega^{(e)}} \varphi_i\varphi_j dx, \quad A_{i,j}^{(e)} = \int_{\Omega^{(e)}} \varphi_i\varphi_j dx$$

- $A_{i,j}^{(e)} \neq 0$ if and only if i and j are nodes in element e (otherwise no overlap between the basis functions)
- All the nonzero elements in $A_{i,j}^{(e)}$ are collected in an *element matrix*
- The element matrix has contributions from the φ_i functions associated with the nodes in element
- It is convenient to introduce a *local numbering* of the nodes in an element: $0, 1, \ldots, d$



Using a reference element $X \in [-1, 1]$

- a *reference cell* in a local reference coordinate system $X \in [-1, 1]$
- a set of *basis functions* $\tilde{\varphi}_r$ defined on the cell
- a correspondence between local and global degree of freedom numbers
- a geometric *mapping* of the reference cell onto to aa cell in the physical domain: $[-1, 1] \;\Rightarrow\; [x_L, x_R]$

The reference cell concept extends to multiple space dimensions!

Integral transformation

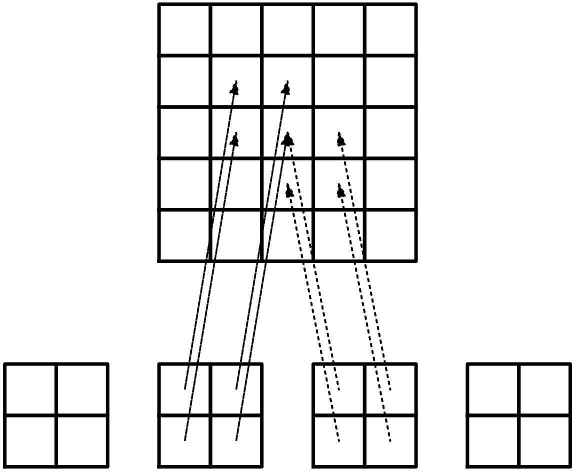
Reference element integration: just change integration variable from x to X. Introduce local basis function

$$\tilde{\varphi}_r(X) = \varphi_{q(e,r)}(x(X))$$

$$\tilde{A}^{(e)}_{r,s} = \int_{\Omega^{(e)}} \varphi_{q(e,r)}(x)\varphi_{q(e,s)}(x)dx = \int_{-1}^{1} \tilde{\varphi}_r(X)\tilde{\varphi}_s(X) \underbrace{\frac{dx}{dX}}_{\det J = h/2} dX = \int_{-1}^{1} \tilde{\varphi}_r$$

$$\tilde{b}^{(e)}_r = \int_{\Omega^{(e)}} f(x)\varphi_{q(e,r)}(x)dx = \int_{-1}^{1} f(x(X))\tilde{\varphi}_r(X) \det J \, dX$$

Illustration of the matrix assembly: regularly numbered P1 elements



Applying FEM to solving PDEs

- Need a *variational formulation* of the PDE problem
  - including integration by parts
- Need care with boundary conditions
- Element-wise computation (via a reference cell) is recommended
  - need to assemble element matrices and vectors

Basic idea

Given a function space $V = \text{span}\{\varphi_0(x), \ldots, \varphi_N(x)\}$, for any PDE

$$\mathcal{L}(u_{\text{e}}) = 0, \quad x \in \Omega$$

we try to find $u(x) = \sum_{j\in\mathcal{I}_s} c_j\varphi_j(x)$ such that residual $R(u)$ is "minimized" (by either a least squares approach or a projection/Galerkin approach)

The Galerkin method

Idea: make R orthogonal to V,

$$(R, v) = 0, \quad \forall v \in V$$

This implies

$$(R, \psi_i) = 0, \quad i \in \mathcal{I}_s$$

$N + 1$ equations for $N + 1$ unknowns $\{c_i\}_{i\in\mathcal{I}_s}$

Integration by parts is an essential ingredient

Rule for multi-dimensional integration by parts:

$$-\int_{\Omega} \nabla \cdot (\alpha(\boldsymbol{x})\nabla u)v \,\mathrm{d}x = \int_{\Omega} \alpha(\boldsymbol{x})\nabla u \cdot \nabla v \,\mathrm{d}x - \int_{\partial\Omega} \alpha \frac{\partial u}{\partial n} v \,\mathrm{d}s$$

Motivation:

- Lowers the order of derivatives
- Gives more symmetric forms (incl. matrices)
- Enables easy handling of Neumann boundary conditions
- Finite element basis functions φ_i have discontinuous derivatives (at cell boundaries) and are not suited for terms with φ_i''

1D example of integration by parts

$$\begin{aligned}\int_0^L u''(x)v(x)dx &= -\int_0^L u'(x)v'(x)dx + [vu']_0^L \\ &= -\int_0^L u'(x)v'(x)dx + u'(L)v(L) - u'(0)v(0)\end{aligned}$$

Dealing with non-zero Dirichlet boundary conditions

- Formal approach: $u(x) = B(x) + \sum_{j\in\mathcal{I}_s} c_j\varphi_{\nu(j)}(x)$
- I_b: set of indices with nodes where u is known

$$B(x) = \sum_{j\in I_b} U_j\varphi_j(x)$$

- Another approach is to first "insist" $u(x) = \sum_j c_j\varphi_j(x)$ and don't include the boundary function $B(x)$, but later "modify" the rows of the linear system that correspond to the nodes that have Dirichlet boundary conditions

FEM and time-dependent PDEs

Overall strategy:

- Finite differencing applied to the time direction first
- Variational formulation then applied in space to the time-discrete problem

Solving nonlinear problems

Main idea:

- Solving a nonlinear problem as a sequence of linearized problems
- Main linearization techniques
    - Picard iterations
    - Newton's method